

Industrial Internet of Things Intrusion Detection System Based on Federated Learning

Teng Fang ^{1,2} and Lina Ge ^{1,2,3}(✉)

¹ School of Artificial Intelligence, Guangxi Minzu University, Nanning, China

² Key Laboratory of Network Communication Engineering, Guangxi Minzu University, Nanning, China

³ Guangxi Key Laboratory of Hybrid Computation and IC Design Analysis, Nanning, China
66436539@qq.com

Abstract. With the rapid development of Industrial Internet of Things (IIoT), it improves the productivity of enterprises, saves the generation cost, and promotes the construction of Industry 4.0, and at the same time, it also increases the attacked surface of IIoT. Traditional centralized Intrusion Detection Systems (IDS) face the dual challenges of data silos and privacy protection. In this paper, we propose a federated learning-based intrusion detection system for IIoT that aims to protect data privacy and improve model performance. We design a deep learning model that combines Bi-directional Long Short-Term Memory Network (Bi-LSTM) and Transformer and incorporates a federated learning framework. Training across multiple distributed nodes is achieved through the federated learning framework. In terms of protecting the data privacy of the participating nodes, we adopt the encryption technique based on Paillier's cipher to ensure the security of the data during the model training process. We validate the respective roles and contributions of Bi-LSTM and Transformer in the model. In addition, we conducted comparative experiments of the proposed model with current state-of-the-art solutions as well as classical centralized models. The experimental results show that our solution can effectively train intrusion detection models with better performance while protecting data privacy.

Keywords: Industrial Internet of Things, Intrusion Detection, Federated Learning, Bi-LSTM, Transformer.

1 Introduction

In the late 20th and early 21st centuries, the fundamental need for industrial networks to be monitored and controlled to ensure the security and reliability of physical devices emerged [1]. Intrusion detection systems (IDS) were then introduced to monitor and analyze industrial network traffic and identify potential attacks [2]. Intrusion detection systems for industrial systems were developed in the mid-2000s, which could understand industrial protocols and identify the normal operation of physical devices [3]. In the late 2000s, with the development of IoT technology, a large number of

industrial Internet of Things (IIoT) devices were integrated into industrial control systems, which required IDS to recognize more diverse data [4].

Randomized IoT technology, communication technology, software technology development, industrial IoT is facing increasing network threats, the attack means are more diverse. The traditional intrusion detection technology, mainly relying on malicious behavior, software feature libraries, or modeling of the normal behavior of the system, the former is difficult to identify new means of attack, the latter false alarm rate is high [5]. And the maintenance of traditional intrusion detection system labor costs are also relatively high, to make the traditional IDS can work properly requires frequent maintenance to update the feature database.

With the continuous development of machine learning, more and more machine learning techniques are integrated into the intrusion detection system, which significantly improves the correctness of the intrusion detection system detection and reduces the false alarm rate of the system. Researchers have collected a large amount of traffic data to train their Decision Tree [6] (DT, Decision Tree), Multilayer Perceptron [7] (MLP, Multilayer Perceptron), Support Vector Machine [8] (SVM, Support Vector Machine), K-Neighbor algorithm [9] (KNN, K-Nearest Neighbor), and Logical Neighbor Algorithm [11] (KNN, K-Nearest Neighbor). Nearest Neighbor), Logistic Regression [10] (LR, Logistic Regression) algorithms, and other models have improved the accuracy and efficiency of IDS to varying degrees.

With the emergence of advanced persistent threats in 2010, attacks became more complex and stealthy, and traditional machine learning reached a bottleneck in improving the accuracy and efficiency of intrusion detection system detection [13]. Researchers found that using deep learning was a better solution, and at this stage, people were not simply understanding traffic as vectors or matrices. A number of ways and methods such as encoders, decoders, attention mechanisms, and some other ways and means began to be used to explore the relationships and links between and among the data itself [12]. When the traffic data is treated as time-series data, techniques such as Recurrent Neural Network (RNN), Long Short-Term Memory Network (LSTM), Transformer, etc. are beginning to be integrated into IDS, which also improves the accuracy and efficiency of IDS detection.

Training a good intrusion detection model, in addition to a good initial model, the most important thing is to have a large amount of real data to support model training. However, enterprises and enterprises even between departments of the same enterprise can not be exchanged between the use of data, which involves issues such as privacy and the right to use. The data silo problem caused by various reasons that the data can not be shared and used is an important reason for restricting the training of high-performance models. In 2017, Google's McMahan [14] and others proposed a federal learning technology that allows the "model to move the data does not move". It can train better intrusion detection models while protecting user privacy.

Intrusion detection model is essentially a multi-classification model to identify normal traffic, injection attack, DoS/DDoS attack, malware, backdoor, etc. from the traffic data. The current research on intrusion detection system for industrial IOT still has some problems: 1. Data sample imbalance, data barriers exist between industrial IOT and industrial IOT due to considerations such as policy and production safety,

which leads to unsatisfactory model training, and the detection accuracy of anomalous attacks needs to be improved; With the emergence of new types of attacks such as model inference, the user's privacy and security can not be adequately safeguarded. For the problem that the detection rate of the intrusion detection model needs to be improved, this paper solves it in two ways, one is to combine the advantages of LSTM and Transformer in processing time-series data to propose a BLT deep learning model, and incorporate the model into the federated learning in order to solve the limitation of the model performance by solving the problem of data silo. In addition paillier passwords are used to secure the communication between the training nodes and the aggregation nodes.

The rest of the paper is organized as follows: section 2 describes the state of the art in research on industrial IoT intrusion detection schemes and federated learning based industrial IoT intrusion detection schemes. Section 3 describes our federated learning based detection scheme, including model architecture, federation learning algorithm, and BLT deep learning model. Section 4 describes our comparison experiments using our solution with today's federated learning solutions and classical machine learning solutions on the NSL-KDD dataset and the UNSW-NB15 dataset, and gives a brief analysis of the corresponding results. Section 5 summarizes our work.

2 Related work

This chapter briefly reviews related research focusing on intrusion detection schemes for industrial IoT and federated learning based intrusion detection schemes.

2.1 Intrusion detection solution for industrial IoT

Intrusion detection is a mechanism for tracking intrusion behaviors in a networked environment, often using a feature selection based approach combined with neural networks for feature learning in artificial intelligence techniques.2017 Y. Liu et al[20] designed a detection model using binary logistic regression to obtain critical network data from the routing layer and used it to analyze sensor behaviors, and achieved an accuracy of their model of between 96% and 100%.In 2018 Wu K H et al [21] proposed a convolutional neural network based process state transition intrusion detection algorithm for industrial control systems that implements a two-stage anomaly detection framework.In 2018 SHONE [22] et al proposed a deep learning technique for intrusion detection using an asymmetric deep autoencoder with unsupervised feature learning as a classification model that showed good performance on benchmark datasets.In 2020, KUKKALA [17] et al. proposed a framework called INDRA using Gated Recursive Unit (GRU)-based Recursive Autoencoder (RAE) for identifying anomalies in automotive embedded systems based on Controller Local Area Network (LAN) buses under different attack scenarios.UPADHYAY [19] et al. proposed a new technique for intrusion system for Industrial IoT proposed an integrated framework that combines feature engineering based preprocessing with machine learning classifier using gradient boosting method for feature selection, the

framework improves the detection rate while reducing the training time. In order to address the problem of data integrity attacks in industrial IoT systems, LIU [15] et al. proposed an attack detection method based on LSTM model which is based on complete time series waveform data from current and voltage sensors to train the model. When the training dataset is large enough, the trained model can recognize incomplete data. ISMAIL [23] et al. investigated power theft cyber attacks in industrial IoT system CPS and proposed a human intrusion detection system based on convolutional neural network and recurrent neural network to deal with such cyber attacks. LIU [14] et al. proposed a deep learning technique based on deep learning technique for industrial Internet of Things (IIoT) anomaly detection proposed a deep anomaly detection (DAD) framework based on federated learning (FL), which enables edge devices to collaboratively train the DAD model and improve the generalization ability of the model. In 2021, IMRANA [24] et al. proposed a bi-directional long and short-term memory network (BiDLSTM) based intrusion detection model to detect different types of attacks on the U2R and R2L attacks with higher detection accuracy than the traditional LSTM network. GE [25] et al. developed a feed-forward neural network model with an embedding layer for multi-classification, where the embedding layer is used to encode the high-dimensional classification features, and, in addition, the high-dimensional features are encoded by using migration learning in order to construct binary classifiers. The above scheme usually preprocesses the network traffic data and then utilizes neural networks to extract the hidden features of the data to accomplish the multi-classification task, thus effectively improving the performance of the intrusion detection model.

2.2 Intrusion detection solution based on federated learning

In recent years, some researchers have applied federated learning to intrusion detection, which solves the “data silo” problem, but also faces issues such as privacy protection and communication cost, which are outlined in this paper. T.D. Nguyen et al [14] proposed a federated IoT self-learning anomaly detection system, which is the first intrusion detection system to employ a federated learning approach for anomaly-based detection. NISHIO [26] et al. proposed a FedCS protocol for mobile edge computing framework, which solves the client selection problem with resource constraints by allowing the server to aggregate as many client updates as possible and accelerating the convergence of the deep learning model, which compared to the original FL protocol shortens the training time. Rong Wang [27] et al. proposed an intrusion detection method based on federated learning and convolutional neural networks, which solves the problem of insufficient amount of data for a single organization. In the same year, RAHMANI [28] et al. proposed an IoT intrusion detection scheme based on federated learning, which protects the privacy of data by performing model local training, but there is no encryption of model parameters in the client communication with the cloud server is vulnerable to model inference attacks. In the same year, ZHAO [33] et al. proposed an effective intrusion detection method based on federated learning-assisted long and short-term memory (FL-LSTM) framework. Simulation results show that the method has higher accuracy and

consistency compared to traditional methods, but it does not protect the model parameters during the communication process. Literature [30] proposes a multi-criteria based federated learning client selection method called FedMCCS, which predicts whether a client is available to perform a federated learning task based on parameters such as CPU, memory, energy, and time and maximizes the number of clients participating in training in each round. Experiments show that the method reduces the number of communication rounds while achieving the desired accuracy. In 2020, LI [18] et al. proposed a federated learning based intrusion detection framework called Deepfed for identifying threats in cyber-physical systems (CPS). A combination of Convolutional Neural Networks (CNNs) and Gated Recurrent Units (GRUs) is used for threat identification and a security protocol based on the Paillier cryptosystem is used to ensure the security of the local and global models during the federated learning training process. In 2020, MAN [20] et al. proposed an intelligent intrusion detection mechanism called FedACNN, which is used to train the models by introducing an attention mechanism to the model training process, which alleviates the communication delay limitation of federated learning to a certain extent achieves the desired accuracy with 50% reduction in the number of communication rounds. In 2021, WANG X D et al [32] proposed a hierarchical federated learning-based anomaly detection for industrial IoT, which utilizes federated learning techniques to build a generalized monitoring model, and then deep reinforcement learning algorithms to train the local model that High throughput, low latency and high accuracy were achieved.

3 This article plan

This subsection describes our proposed scheme, the core of this scheme is to propose a BLT deep learning model for incorporation into a federated learning framework, and design a federated learning algorithm based on the paillier cipher for coordinating the completion of model training. There are three types of nodes in my scheme, namely Data Center (DC) nodes, Control Center (CC) nodes which can also be called aggregation nodes and DataConcentrator (DCT) nodes which can also be called training nodes. The three constituent federated learning models are shown in Figure 1:

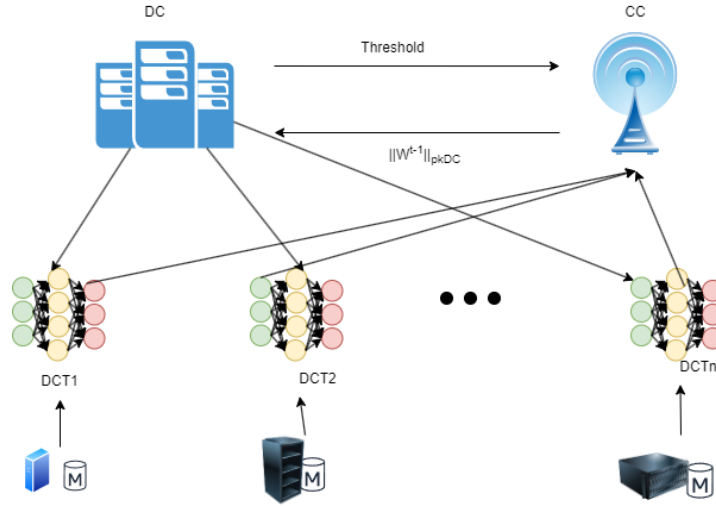


Fig. 1. The BLT-FL model.

The content of the communication between DC, DCT_i and CC can be briefly described as shown in Figure 2

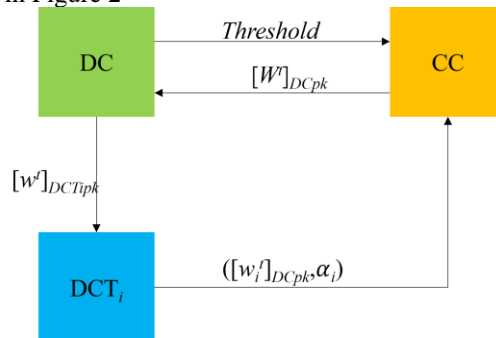


Fig. 2. BLT-FL data communication sketch

1) Data Center: DC is responsible for determining the intrusion detection learning task and sending the initialized model parameters encrypted with the public key of each training node to the corresponding data concentrator DCTs (i.e., training nodes) in turn, and also sends the threshold *Threshold* to screen the participants of the DCTs to the CC. At the same time, the DC will receive the encrypted global model parameters from the CC after aggregation, decrypt them and use its own data to Perform invasive model training, and then proceed to the next round of iterative updating

2) Data Concentrator: DCT as the edge node of IIOT is responsible for monitoring the network status and collecting network traffic from smart meters at regular intervals. In order to be able to train the intrusion detection model, this paper assumes that each

DCT has enough storage capacity to save the collected historical data and at the same time has the computational capability to perform intrusion detection locally. In the local model training phase, each DCT first obtains the intrusion detection model with parameters from the DC for local updating, and calculates the similarity between the local model and the global model: then the encrypted parameters of the local model and the similarity metric are sent to the CC.

3) Control Center: CC receives the model parameters and locally updated similarity metrics transmitted from DCT. In order to obtain the global optimal intrusion detection model more efficiently, CC is mainly responsible for filtering the received local models, securely aggregating the better local models according to the filtering threshold set by DC, and then sending the aggregated global parameters to DC for subsequent iterative training.

The subsequent contents of this paper will explain in detail the deep learning models and federated learning algorithms we trained in turn.

3.1 BLT model

As shown in Figure 3 The deep learning model (BLT) used in this paper consists of six parts: the Bi-LSTM module, the Transformer module, the flatten layer, the Droupt layer, the Dense layer, and the softmax layer.

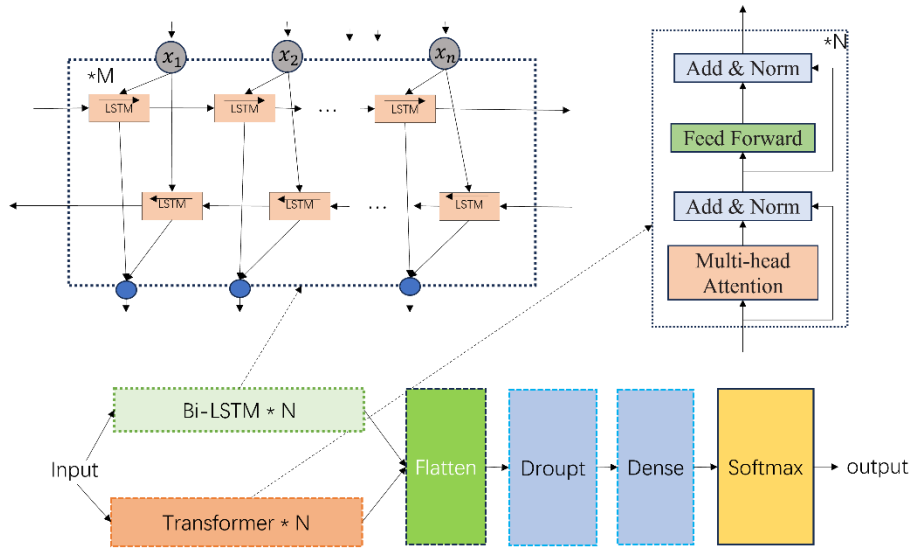


Fig. 3. BLT deep learning model

The number of submodules in the Bi-LSTM module and the Transformer module will be dynamically adjusted in specific applications. A brief description of each component follows in this subsection.

BiLSTM layer: in order to capture long-range dependent features, x_t is fed into the BiLSTM model, which consists of LSTM modules connected in two directions with multiple shared weights. At each time step, the output of the BiLSTM module will be controlled by a forgetting gate (ft), an input gate (it), an output gate (ot) and a cell state update jointly. Each gate is represented by the output $bt-1$ of the previous module and the input x_t at the current moment, and the three gates work together to fulfill the selection of the attribute information, forgetting, and the cell state update. At time step t , the input x_t is feature extracted using the forward part of the BiLSTM module as in Equation 1:

$$\text{Forward LSTM} \Rightarrow \begin{cases} i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ q_t = \tanh(W_q \cdot [h_{t-1}, x_t] + b_q) \\ o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\ c_t = f_t * c_{t-1} + i_t * q_t \\ b_t = o_t * \tanh(c_t) \end{cases} \quad (1)$$

At time step t , x_t is feature extracted using the inverse part of the BiLSTM module as in Equation 2:

$$\text{Backward LSTM} \Rightarrow \begin{cases} i_t = \sigma(W_i \cdot [h_{t+1}, x_t] + b_i) \\ f_t = \sigma(W_f \cdot [h_{t+1}, x_t] + b_f) \\ q_t = \tanh(W_q \cdot [h_{t+1}, x_t] + b_q) \\ o_t = \sigma(W_o \cdot [h_{t+1}, x_t] + b_o) \\ c_t = f_t * c_{t+1} + i_t * q_t \\ b_t = o_t * \tanh(c_t) \end{cases} \quad (2)$$

where σ is the sigmoid activation function; \tanh is the hyperbolic tangent function; $*$ is the element multiplication operation, i_t is the selection operation of input information to control the input process of information, and f_t is the forgetting operation of the information that needs to be forgotten in the previous module to control the process of information forgetting. c_t is used to judge which information should be stored in the current cell state, and to complete the control of information storage to complete the control of the information. o_t is the output gate to select the output information and control the output information.

The final output feature vector u_t of the Bi-LSTM module at time step t is:

$$u_t = [\text{Forward LSTM}, \text{Backward LSTM}] \quad (3)$$

As shown in Figure 3, the encoder part of Transformer is used in this paper. The input x_t is divided into numeric records and type records, both of which use column embedding to embed each $x_{t,i}$ into the m -dimension. The process can be expressed as follows: $E(x_t)$ As shown in Figure 3, this paper uses the encoder part of the Transformer. The input x_t is divided into numeric records and type records, and each $x_{t,i}$ is embedded into the m -dimension using column embedding. The process can be expressed as $E(x_t) = \{e_1(x_{t,1}), \dots, e_n(x_{t,n})\}$, $e_i(x_{t,i}) \in \mathbb{R}^{1 \times m}$, $E(x_t)$ is the column embedding of x_t . Then, the embedded classification features $E(x_t)$ are transformed into inputs to the Transformer module $I = (e_1(x_{t,1}), \dots, e_n(x_{t,n}))^T \in \mathbb{R}^{n \times m}$ and fed into the first layer of the Transformer module, where the output of the first

Transformer layer is fed into the next Transformer layer, and so on. A transformer layer contains the Multihead Attention Layer, Add&Norm, Feed Forward, and Add&Norm. at the Multihead Attention Mechanism layer there are three learnable parameters $W_Q \in \mathbb{R}^{m \times d_k}$, $W_K \in \mathbb{R}^{m \times d_k}$, and $W_V \in \mathbb{R}^{m \times d_v}$. The input I of the module is projected into three matrices $K \in \mathbb{R}^{n \times d_k}$, $Q \in \mathbb{R}^{n \times d_k}$ and $V \in \mathbb{R}^{n \times d_v}$.

$$Q = IW_Q, K = IW_K, V = IW_V \quad (4)$$

Where m is the number of features input to the Transformer, and d_v , d_k are the hidden dimensions of the multi-head attention layer. Next the output of the multi-head attention layer is computed:

$$\text{Attention}(K, Q, V) = A \cdot V \quad (5)$$

Where $A = \text{softmax}((QK^T)/\sqrt{d_k})$ is the attention matrix that captures the similarity between input embeddings. Then the output of the attention head is projected back to the input dimension n through a feed-forward network. Consequently, through successive transformer layers, the embedded $E(x_t)$ will be transformed into contextual embeddings, namely, the relationship between the categorical features can be greatly captured in the output of the last transformer layer.

Then Bi-LSTM is connected with the output of Transformer as the input of Flatten layer. And then it goes through the Dropout layer, Dense layer, and softmax layer in turn to arrive at the result of multi-classification.

3.2 Federated Learning Algorithm Based on Paillier Cipher

It is assumed that there is a trusted key generation center used to generate key pairs for DC and DCT to encrypt and protect model parameter transmission. Specifically, generate a pair of asymmetric keys for DC for Paillier homomorphic encryption (sk_{DC}, pk_{DC}), DC retains its own private key sk_{DC} for decrypting the aggregated model parameters, pk_{DC} broadcasts the public key and uses it to encrypt the local model parameters transmitted by DCT to CC. In addition, each data concentrator also holds a pair of asymmetric keys (sk_i, pk_i), private sk_i by the i th participant DCT_i ; reserved, used to decrypt the global model parameters broadcast by the DC, and the public key is used to encrypt pk_i the global model parameters broadcast to each data center DCT_i .

Model parameter initialization.

DC determines the intrusion detection model and broadcasts the global model parameters. Since the trained neural network model usually carries data set information, such as location information, data set size, etc., in order to prevent malicious eavesdroppers from inferring the intercepted model parameters. If the attack obtains the above private information, it needs to be encrypted and broadcast. First, the data center uses local data to train the intrusion detection model to obtain the model parameters w^{init} ; then uses $DCT_i (i \in [1, m])$ the public key of each participant pk_i ; encrypts it w^{init} to obtain the encrypted model parameters $[w^{init}]_{pk_i}$ and broadcasts them to all participants DCT_i ; finally, the DC performs intrusion detection

in each round of iterations. Before training, determine the similarity threshold Threshold for screening participants, and send the Threshold to CC for screening local models .

Local model update.

Assume that each participant DCT_i has its local network traffic data set $D_i = \{x_j, y_j\} (j \in [1, N_i])$, and its data size $|D_i| = N_i (i \in [1, m])$, m , is the number of participants. Because this article uses the convolutional neural network model to perform multi-class detection of network attacks, the loss function selected is the multi-class cross entropy function, as shown in formula (6)

$$f_{loss} = - \sum_{k=1}^K y_k \log(\hat{y}_k) \quad (6)$$

Among them, y_k represents k th the real label corresponding to the t th category, \hat{y}_k is k th the predicted label of the t th category, and represents K a total of categories. In the $t+1$ round of the global iteration, after each participant DCT receives the global model parameters broadcast by DC $[[w^t]]_{pk_i}$, it uses its own private key sk_i ; decrypts it to obtain the global model parameters w^t , uses its own private data set to update the local model, and uses the Adam gradient descent algorithm for optimization, i.e.

$$w_i^{t+1} \leftarrow w^t - \lambda \nabla f(w^t; j) \quad (7)$$

Among them, λ is the learning rate of model training and j is the sample serial number.

Local model similarity calculation.

In machine learning, Euclidean distance measures the direct space between points, whereas cosine distance evaluates vector similarity based on angle. For federated training, cosine similarity is favored for its alignment with the global model's direction, emphasizing orientation over magnitude.

To assess model convergence, we address the initial challenge of unknown current global parameters by using the previous round's parameters as a proxy. This approach is based on the assumption of steady model convergence, which minimizes the variation between consecutive updates. Specifically, given the sum of global model parameters of two consecutive rounds of round t and round $t+1$ w^t , w^{t+1} the normalized difference measure between the two is defined as :

$$\Delta w_t = \frac{\|w^{t+1} - w^t\|}{\|w^t\|} \quad (8)$$

Where $\|\cdot\|$ is the Euclidean norm of a given vector. Intuitively, the larger its value, the greater the difference between two consecutive global models. ASAD et al. [16] experimentally proved that the difference between the model parameters of two consecutive iterations is approximately 0.05. Therefore, this article can use the global model parameters of the previous iteration as a perfect substitute for the global model parameters of this time. In order to facilitate calculation, each participating DCT needs to vectorize the model parameters. Assume that the local model parameters of the current participant $w_i^{t+1} = [l_1, l_2, \dots, l_N]$ and the global model parameters in the previous iteration are $w_i^t = [g_1, g_2, \dots, g_N]$, N is the dimension of the vector, and then their cosine similarity measure is calculated through formula, that is

$$\begin{aligned}
& \text{cos_similarity}(w_i^{t+1}, w^t) \\
= & \cos(w_i^{t+1}, w^t) = \frac{w_i^{t+1} w^t}{\|w_i^{t+1}\| \cdot \|w^t\|} \\
= & \frac{\sum_{j=1}^N (l_j \times g_j)}{\sqrt{\sum_{j=1}^N (l_j)^2} \times \sqrt{\sum_{j=1}^N (g_j)^2}}
\end{aligned} \tag{9}$$

Since the range of the cosine value is $[-1, 1]$, when the cosine value is closer to 1, the closer the two model parameters are. When the cosine value is close to -1, it means that they are developing in opposite directions. Since a similarity value less than 0 is inconsistent with reading habits, this article normalizes the similarity measure to the range $[0, 1]$, and rewrites formula (9), that is

$$\begin{aligned}
\alpha_i &= \text{cos_similarity}(w_i^{t+1}, w^t) \\
&= 0.5 \times \cos(w_i^{t+1}, w^t) + 0.5
\end{aligned} \tag{10}$$

In order to facilitate writing and subsequent calculations, this article uses α_i as the similarity measure value of the participant ;.DCT_i

In summary, through formula (7) and formula (8), the local model parameters and similarity measurement value $\alpha_1, \alpha_2, \dots, \alpha_m$ of each DCT are obtained $w_1^{t+1}, w_2^{t+1}, \dots, w_m^{t+1}$. The local model parameters are encrypted using the public key pk_{pc} to generate model parameter ciphertext $[w_1^{t+1}]_{pk_{DC}}, [w_2^{t+1}]_{pk_{DC}}, \dots, [w_m^{t+1}]_{pk_{DC}}$. And the model parameter ciphertext and the corresponding similarity measurement value are transferred to CC for filtering local models and secure aggregation of global models.

Global aggregation.

After CC receives the encrypted model parameters and corresponding similarity measures α_i uploaded by the participant DCT $[w_i^{t+1}]_{pk_{DC}}$, it first filters the local model. At that time , this article discards the local update.

When $\text{cos_similarity}(w_i^{t+1}, w^t) < \text{Threshold}$ Therefore, the local update that participates in the global aggregation with better performance $[w_1^{t+1}]_{pk_{DC}}, [w_2^{t+1}]_{pk_{DC}}, \dots, [w_M^{t+1}]_{pk_{DC}}$ ($M < m$), is the number of participants in the final global aggregation. In order to not lose generality and facilitate the subsequent encryption calculation operations, this article will α_i treat it as a positive integer (although α_i it is a decimal, it can be converted into a positive integer by multiplying it by the integer power of 10, and then divide the aggregated result by The same multiple is sufficient). Then CC executes formula to perform a weighted aggregation operation to obtain the global model parameters in the aggregated ciphertext state, namely

$$[w^{t+1}]_{pk_{DC}} \leftarrow \prod_{i=1}^M [w_i^{t+1}]_{pk_{DC}}^{\alpha_i} \tag{11}$$

According to the additive and multiplicative properties of Paillier homomorphic cryptography, formula (6) can be further calculated, that is

$$\begin{aligned}
& \prod_{i=1}^M [w_i^{t+1}]_{pk_{DC}}^{\alpha_i} \\
&= \prod_{i=1}^M [\alpha_i w_i^{t+1}]_{pk_{DC}} \\
&= [\alpha_1 w_1^{t+1}]_{pk_{DC}} [\alpha_2 \cdot w_2^{t+1}]_{pk_{DC}} \cdots [\alpha_M w_M^{t+1}]_{pk_{DC}} \quad (12) \\
&= [\alpha_1 w_1^{t+1} + \alpha_2 w_2^{t+1} + \cdots + \alpha_M w_M^{t+1}]_{pk_{DC}} \\
&= [\sum_{i=1}^M \alpha_i w_i^{t+1}]_{pk_{DC}}
\end{aligned}$$

Therefore, formula (7) can be transformed into formula (8)

$$[w^{t+1}]_{pk_{DC}} \leftarrow [\sum_{i=1}^M \alpha_i w_i^{t+1}]_{pk_{DC}} \quad (13)$$

Finally, CC transmits the encrypted global model parameters $[w^{t+1}]_{pk_{DC}}$ to the data center DC .

Decrypt.

DC receives the global model parameters encrypted by CC and $[[w^{t+1}]]_{pk_{DC}}$ decrypts them w^{t+1} with its own private key , and obtains the global model parameters of this round through formula (4) sk_{DC} . At this point, this round of iterative training ends and then the next round of iterations proceeds.

4 Simulation evaluation

In this section, we conduct a number of experiments to evaluate the performance of our proposed BLT-FL scheme. First, we give the experimental setup, including the environment settings, data resource description and partitioning, baseline study and performance metrics.

4.1 Experimental setup

The proposed federated learning framework was implemented in Python using TensorFlow (TensorFlow is an open-source library used in Python for deep learning applications) and the designed deep learning model is implemented using Keras (<https://keras.io/api/>, accessed on 20 Feb 2024) API. The proposed framework is implemented and evaluated using Python 3.0 on a MacBook pro having an Apple M1 Pro chip with 10-core CPU and 16-core GPU, 16 GB RAM, and 1TB SSD.

4.2 Dataset description

The UNSW-NB15 dataset is a cyber intrusion dataset released in 2015 by the Center for Cyber Security at the University of New South Wales, Australia. The dataset contains real cyber-attack data from the period of 2011 to 2015, including 9 attack types and a total of 156 features. The features of the UNSW-NB15 dataset are as follows:

1) The dataset contains a large amount of real cyber-attack data, including 9 common attack types.

2)The dataset has a large number of features that can comprehensively characterize network traffic.

3)The dataset is balanced in terms of attack categories and can effectively evaluate the performance of intrusion detection models.

The **NSL-KDD** dataset is a network intrusion dataset released by Carnegie Mellon University in 1999. The dataset contains real network attack data from the period of 1998 to 1999, including 41 attack types with a total of 41 features. The features of the NSL-KDD dataset are as follows:

1)The dataset contains a large amount of real network attack data, including 41 common attack types.

2)The number of features in the dataset is small, but it can still effectively characterize network traffic.

3) The dataset is unbalanced in terms of attack categories, with a high percentage of “normal” attack categories, which affects the performance of the intrusion detection model.

4.3 Data set processing

The original dataset's character-based labels are numerically encoded using Sklearn's One Hot Encoder for one-hot encoding, mapping discrete labels to Euclidean space for more accurate feature calculations.

Next, the continuity features are normalized. This article uses linear normalization. The formula is as follows:

$$x_{scale} = \frac{x-x_{min}}{x-x_{max}} \quad (14)$$

Among them, x is the value before normalization, x_{scale} is the value after normalization, x_{max} and x_{min} refers to the maximum boundary and minimum boundary of the data that need to be normalized.

4.4 Data results and analysis.

The metrics used to assess the proposed BLT-FL framework are classification Accuracy, Precision, Recall, and F1-Score. True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN), are used to express these metrics, respectively, where: TP: Specify the count of attack requests rightly predicted as attack; TN: Specify the count of benign samples rightly predicted as benign; FP: Count of benign requests falsely predicted as attack; FN: Count of attack requests falsely predicted as benign.

1) Accuracy: It is the percentage of the right prediction of attack and benign requests

$$Accuracy = \frac{(TP+TN)}{(TP+TN+FP+FN)} \quad (15)$$

2) Recall: It is defined as the ratio of right prediction of attack to the all observations in actual class

$$Recall = \frac{TP}{(TP+FN)} \quad (16)$$

- 3) Precision: It is the ratio of right prediction of benign request to the entire predicted benign requests.

$$\text{Precision} = \frac{TP}{(TP+FP)} \quad (17)$$

- 4) F1-Score: This score is the measure of test's accuracy and calculated using Precision and Recall.

$$\text{F1-Score} = 2 * \left(\frac{(\text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})} \right) \quad (18)$$

Our experiments using the NSL-KDD dataset compared centralized training with federated deep learning BLT, with results in Figure 2 indicating BLT's federated approach outperforms traditional methods in accuracy.

Experiment 1 Cross-Comparison Experiments

We conduct the first series of experiments to compare the performance of our proposed intrusion detection model with some of the state-of-the-art research, including DeepFed [18], AMCNN-LSTM [29], and CFL-IDS [31]. In addition, we compare the performance of the developed intrusion detection model with a local intrusion detection model constructed by each industrial agent and an ideal intrusion detection model constructed by a centralized entity over all data resources. The experimental results are shown in Figure 3.1 and Table 3.1:

DeepFed is a federated learning-based intrusion detection model proposed by Beibei Li et al [18] based on Convolutional Neural Network (CNN) and Gated Recurrent Unit (GRU).

Liu et al [29] came out with a federated learning (FL)-based deep anomaly detection framework that introduces a hybrid model combining a convolutional neural network with an attention mechanism and a long and short-term memory network (AMCNN-LSTM).

CFL-IDS [31] is Shan Y et al [31] et al. constructed an IIoT intrusion detection framework (CFL-IDS) based on the clustering of edge nodes (ENs) local model evaluation metrics (EMs).

We ran 8 sets of training and used 4 schemes on both datasets split into 30 clients to train until convergence, and the experimental results are shown in Table1. where time is the time spent on a single round of training.

As shown in Table 1, our proposed solution, BLT-FL, is excellent in terms of the item evaluation metrics are higher than other solutions.

Table 1. Performance of different methods on different data sets

Methods	UNSW-NB15					
	accuracy	Precision	Recall	F1-score	FPR	Time(s)
DeepFed[18]	0.73136	0.71831	0.73756	0.7278	0.27453	70.13
AMCNN-LSTM[29]	0.87438	0.86589	0.88084	0.8733	0.13187	67.43
CFL-IDS[31]	0.91341	0.90718	0.91862	0.91286	0.09168	85.31

BLT-FL	0.93258	0.92184	0.94207	0.93185	0.07652	61.09
NSL-KDD						
Methods	accuracy	Precision	Recall	F1-score	FPR	Time(s)
DeepFed[18]	0.68074	0.67068	0.68445	0.67749	0.32283	75.32
AMCNN-LSTM[29]	0.76464	0.75763	0.76841	0.76298	0.23902	83.64
CFL-IDS[31]	0.84683	0.83538	0.85496	0.84505	0.16094	89.39
BLT-FL	0.86277	0.83049	0.8514	0.84082	0.16545	75.06

Figure 3.1 illustrates the change in accuracy during model training, where (a) shows the change in accuracy for the four solutions on the UNSW-NB15 dataset and (b) shows the change in accuracy for the four solutions on the NSL-KDD dataset. It is not difficult to find that our proposed BLT-FL solution converges fast and performs well through the two patterns.

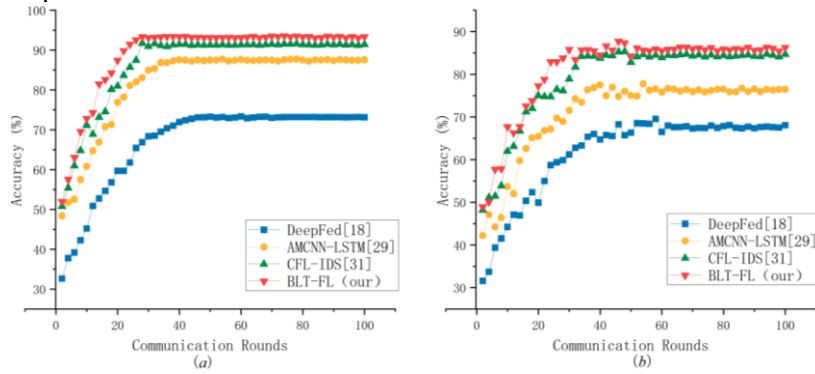


Fig. 4. Comparison of the training accuracy of the four schemes on different datasets

Experiment 2 Longitudinal Comparison Experiments

This series of experiments compares federated learning schemes and machine learning schemes that have been well established for a long time, among them CNN, KNN, LR, SVM, MPL, FedAvg, FedProx. The federated learning schemes all use 10 PCs with the same performance as training nodes to train intrusion detection models.

Table 2. Comparison of performance indicators of different programs

	accuracy	Precision	Recall	F1-score	FPR
LR	0.5524	0.5561	0.552	0.554	0.4472
KNN	0.599	0.5905	0.6007	0.5956	0.4027
SVM	0.7155	0.7238	0.712	0.7178	0.2809
MPL	0.6526	0.6473	0.6543	0.6508	0.349
CNN	0.89	0.8903	0.8898	0.89	0.1098

FedAvg	0.8262	0.8532	0.8095	0.8308	0.1551
FedProx	0.8561	0.8718	0.8452	0.8583	0.1324
BLT-FL	0.9442	0.9318	0.9555	0.9435	0.0665

5 Summary

In this paper, we propose a joint learning-based intrusion detection system (BLT-FL) for industrial IoT, which aims to address the shortcomings of traditional centralized intrusion detection systems in terms of privacy protection and model recognition rate. Our system integrates Bi-LSTM and Transformer models for network traffic analysis to improve the accuracy of intrusion detection. The system utilizes joint learning techniques to achieve collaborative global model training across participants without sharing data, which both protects privacy and most significantly improves the accuracy of model detection. Experimental results show that our BLT-FL performs well in terms of detection rate, false alarm rate, and model convergence speed compared to existing centralized training models and other federated learning models. In addition, we introduce Paillier homomorphic encryption to enhance the security of model parameters during transmission. In order to improve the system efficiency, we design a filtering mechanism based on model similarity, which can filter out the local models that contribute more to the global model in each round of training, thus reducing unnecessary communication and computational resource consumption. Experiments validate the effectiveness of our framework, which not only improves the accuracy of intrusion detection while ensuring model convergence, but also significantly reduces the communication bandwidth requirement while ensuring the privacy and security of user data. The FL-IDS proposed in this paper provides a secure, efficient, and privacy-preserving solution for intrusion detection in IIoT environments, which is expected to play an important role in real-world industrial IoT security protection. Future work will explore more efficient aggregation algorithms and privacy-preserving techniques to further improve the performance of intrusion detection systems.

Acknowledgement. This work was supported by the National Natural Science Foundation of China under Grant 61862007, and Guangxi Natural Science Foundation under Grant 2020GXNSFBA297103.

References

1. Sun Yian, Jing Ke, Wang Yizhou. Research on Industrial Control System Security Network Protection [J]. *Information Security Research*, 2017, 3(02): 171-176.
2. Hu Y, Yang A, Li H, et al. A survey of intrusion detection on industrial control systems[J]. *International Journal of Distributed Sensor Networks*, 2018, 14(8): 1550147718794615.
3. Sun M, Lai Y, Wang Y, et al. Intrusion detection system based on in-depth understandings of industrial control logic[J]. *IEEE Transactions on Industrial Informatics*, 2022, 19(3): 2295-2306.

4. Boyes H, Hallaq B, Cunningham J, et al. The industrial internet of things (IIoT): An analysis framework[J]. *Computers in industry*, 2018, 101: 1-1
5. Bhati N S, Khari M, García-Díaz V, et al. A review on intrusion detection systems and techniques[J]. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 2020, 28(Supp02): 65-91.
6. Quinlan J R. *C4. 5: programs for machine learning*[M]. Elsevier, 2014.
7. Riedmiller M, Lenczner A. Multi layer perceptron[J]. *Machine Learning Lab Special Lecture*, University of Freiburg, 2014, 24.
8. Pisner D A, Schnyer D M. *Support vector machine*[M]//*Machine learning*. Academic Press, 2020: 101-121.
9. Keller J M, Gray M R, Givens J A. A fuzzy k-nearest neighbor algorithm[J]. *IEEE transactions on systems, man, and cybernetics*, 1985 (4): 580-585.
10. Hosmer Jr D W, Lemeshow S, Sturdivant R X. *Applied logistic regression*[M]. John Wiley & Sons, 2013.
11. Mishra P, Varadharajan V, Tupakula U, et al. A detailed investigation and analysis of using machine learning techniques for intrusion detection[J]. *IEEE communications surveys & tutorials*, 2018, 21(1): 686-728.
12. Namasudra S, Dhamodharavadhani S, Rathipriya R. Nonlinear neural network based forecasting model for predicting COVID-19 cases[J]. *Neural processing letters*, 2023: 1-21.
13. B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: *Artificial Intelligence and Statistics*, PMLR, 2017, pp. 1273–128
14. Nazir, Anjum, and Rizwan Ahmed Khan. "A novel combinatorial optimization based feature selection method for network intrusion detection." *Computers & Security* 102 (2021): 102164.
15. Liu, Jingmei, Yuanbo Gao, and Fengjie Hu. "A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM." *Computers & Security* 106 (2021): 102289.
16. Melis, Luca, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. "Exploiting unintended feature leakage in collaborative learning." In *2019 IEEE symposium on security and privacy (SP)*, pp. 691-706. IEEE, 2019.
17. Kukkala V K, Thiruloga S V, Pasricha S. INDRA: Intrusion detection using recurrent autoencoders in automotive embedded systems[J]. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2020, 39(11): 3698-3710.
18. Li, Beibei, et al. "DeepFed: Federated deep learning for intrusion detection in industrial cyber-physical systems." *IEEE Transactions on Industrial Informatics* 17.8 (2020): 5615-5624.
19. Upadhyay D, Manero J, Zaman M, et al. Gradient boosting feature selection with machine learning classifiers for intrusion detection on power grids[J]. *IEEE Transactions on Network and Service Management*, 2020, 18(1): 1104-1116.
20. Liu, Yufei, and Dechang Pi. "A novel kernel SVM algorithm with game theory for network intrusion detection." *KSII Transactions on Internet and Information Systems (TIIS)* 11.8 (2017): 4043-4060.
21. Wu, Kehe, Zuge Chen, and Wei Li. "A novel intrusion detection model for a massive network using convolutional neural networks." *Ieee Access* 6 (2018): 50850-50859.
22. Shone, Nathan, et al. "A deep learning approach to network intrusion detection." *IEEE transactions on emerging topics in computational intelligence* 1 (2018): 41-50.

23. Ismail M, Shaaban M F, Naidu M, et al. Deep learning detection of electricity theft cyber-attacks in renewable distributed generation[J]. *IEEE Transactions on Smart Grid*, 2020, 11(4): 3428-3437.
24. Imrana, Yakubu, et al. "A bidirectional LSTM deep learning approach for intrusion detection." *Expert Systems with Applications* 185 (2021): 115524.
25. Ge, Mengmeng, et al. "Towards a deep learning-driven intrusion detection approach for Internet of Things." *Computer Networks* 186 (2021): 107784.
26. Nishio, Takayuki, and Ryo Yonetani. "Client selection for federated learning with heterogeneous resources in mobile edge." In *ICC 2019-2019 IEEE international conference on communications (ICC)*, pp. 1-7. IEEE, 2019.
27. Wang Rong, Ma Chunguang, and Wu Peng. "Intrusion detection method based on federated learning and convolutional neural network." *Information Network Security* 4 (2020): 47-54.
28. Rahman S A, Tout H, Talhi C, et al. Internet of things intrusion detection: Centralized, on-device, or federated learning?[J]. *IEEE Network*, 2020, 34(6): 310-317.
29. Liu Y, Garg S, Nie J, et al. Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach[J]. *IEEE Internet of Things Journal*, 2020, 8(8): 6348-6358.
30. AbdulRahman S, Tout H, Mourad A, et al. FedMCCS: Multicriteria client selection model for optimal IoT federated learning[J]. *IEEE Internet of Things Journal*, 2020, 8(6): 4723-4735.
31. Shan Y, Yao Y, Zhou X, et al. CFL-IDS: An Effective Clustered Federated Learning Framework for Industrial Internet of Things Intrusion Detection[J]. *IEEE Internet of Things Journal*, 2023.
32. Wang X, Garg S, Lin H, et al. Toward accurate anomaly detection in industrial internet of things using hierarchical federated learning[J]. *IEEE Internet of Things Journal*, 2021, 9(10): 7110-7119.
33. Zhao R, Yin Y, Shi Y, et al. Intelligent intrusion detection based on federated learning aided long short-term memory[J]. *Physical Communication*, 2020, 42: 101157.